APPENDIX 3

Hcs Logging Support

All log records are classified using an ASN.1 Object Id.

Standard Log record ASN.1 Type Classification:

1.x - All log records

1.x.0 - NonCritical

1.x.0.0 - Trace output

1.x.0.1 - Progress

1.x.1 - Error

1.x.2 - Warning

Log records are composed of four fields of information: type, time, creator, information. Type and time are ASN.1 ids while the creator can be either an ASN.1 HcsResource instance id or a text string. The information field is the actual text generated by the logging component.

The are various components in the system that are responsible for the collecting, storage, and possible forwarding of log records. All forms of this component are modeled by the HcsLogFacility class.

HcsLogFacility class

The HcsLogFacility class is an HcsResource derivations who's purpose is to provide storage and forwarding at various levels in the system. There are currently two implementations: the Local Log Facility, and the Central Log Facility. The HcsLogFacility interface is:

```
class HcsLogFacility : public HcsResource
{
public:

        virtual HRESULT putRecord ( char* typeAsn1IdStrPtr,
                                    char* timeAsn1StrPtr,
                                    char* idStrPtr,
                                    char* recordPtr ) = 0;
}
```

The Local Log Facility

An Local Log Facility (LLF) instance exists in every node in the system. The purpose of this HcsLogFacility implementation is to accept all

log records from various HCS components on that node and to buffer them in a large circular on disk until they can be delivered to the Central Log Facility (CLF). The intent is that a node could survive for some time if the CLF were to be unavailable. One of the configuration parameters passed to each Local HcsLogFacility instance will be the resource name of the HcsLogFacility through which that instance is to forward its records. The current thinking is that this will be the name of the CLF but this is not an architectural requirement. In other words there could be intermediate levels of HcsLogFacilities placed in the system. In fact on LLF could be configured to forward its records to another LLF. After all a HcsLogFacility is a HcsLogFacility.

The Central Log Facility

There will be one instance of the Central Log Facility (CLF) instance in the entire system. The purpose of this HcsLogFacility implementation is to accept all log records from various HCS components within the entire system and to provide for the final storage of this information. The intent is that this facility provide for the long term, off line, archival of the entire HCS system log. In addition to this function, it will also provide for the on-line viewing of some portion of the log.

How log records are written

So the question is: how do HCS components log there information? To provide a standard and safe access to the HcsLogFacilites, there is a class family provided. The family is based at a class called HcsLogPort. This class implements the common behavior of all types of LogPorts and is functional on its own.

Derived from the HcsLogPort are the HcsResourceLogPort and HcsLogFacilityPort classes. The HcsResourceLogPort provide easy support for the resource developer while the HcsLogFacilityPort provides direct access to HcsLogFacilites. In general, the first is used by Resource derivations while the second would be used by resource servers and system components.

```
class HcsLogPort
{
public:
    /* Global Constant AsnObjectIds for all defined log record types --
SEE INCLUDE FOR TYPE LIST */
    #include <HcsLogRecTypes.h>

public:
    HcsLogPort ( char* idStrPtr ) ;
```

```
        virtual ~HcsLogPort ( void );
        void put ( const Asn1ObjectId* typePtr, char* formatStrPtr, ... ) ;
        inline char* getIdStrPtr ( void ) { return( myIdStrPtr ); };
        BOOL setIdStr ( char* newIdStrPtr );

        /* Methods to control log filtering */
        void enable ( char* asn1IdOfLogRecTypeToEnablePtr );
        void disable ( char* asn1IdOfLogRecTypeToDisablePtr );
        BOOL isEnabled ( char* asn1IdOfLogRecTypeToChkPtr );
        BOOL isEnabled ( const Asn1ObjectId* logRecTypeToChkPtr );
        inline HRESULT dumpFilterState ( TextSink* toSinkPtr, int
maxLineSize )
                { return( myFilter.dumpState( toSinkPtr ) ); };
        inline void resetFilter ( void ) { myFilter.reset(); };


        void forwardRec ( const Asn1ObjectId* typePtr, char* timeAsn1StrPtr,
char* idStrPtr, char* recordPtr ) ;
                                /* used to forward from one
HcsLogPort implementation to another */

protected:
        /* Derivation Interface */
```

```
        /* Defaults to writing errors only to the NT SystemLog */
        virtual void dumpRec( const Asn1ObjectId* typePtr, char*
timeAsn1StrPtr, char* idStrPtr, char* recordPtr ) ;

private:
        void putOver ( void ) ;
        HcsLogPort ( void ) ;
        void* operator new ( size_t size ) ;

private:
        char*           myIdStrPtr;
        HANDLE          myHandle;
        DWORD           myIdEvent;
        Asn1Set         myFilter;

public:
        /* Grand fathered Methods */
        void setTrace ( BOOL to );
        void setProgress ( BOOL to );
        void setWarning ( BOOL to );
        BOOL isTraceEnabled ( void );
        BOOL isProgressEnabled ( void );
        BOOL isWarningEnabled ( void );
```

```
        void putError ( char* formatStrPtr, ... ) ;
        void putWarning ( char* formatStrPtr, ... ) ;
        void putTrace ( char* formatStrPtr, ... ) ;
        void putProgress ( char* formatStrPtr, ... ) ;
        void putMaintFailure ( char* formatStrPtr, ... ) ;
};

/* class used by resource implementors */
class HcsResourceLogPort : public HcsLogPort
{
public:
        HcsResourceLogPort ( HcsResourceImp* ownerPtr, char* idStrPtr );
        virtual ~HcsResourceLogPort ( void );

protected:
        /* routes all output through the owning HcsResourceImp's
putLogRecord private support method */
        void dumpRec( char* typeAsn1IdStrPtr, char* timeAsn1StrPtr,  char*
idStrPtr, char* recordPtr );

private:
        HcsResourceLogPort ( void );
        void* operator new ( size_t size );
```

```
};


/* class used by non resource centric implementations */
class HcsLogFacilityPort : public HcsLogPort
{
public:
        HcsLogFacilityPort ( char* idStrPtr );
        HcsLogFacilityPort ( HcsLogFacility* facilityPtr, char* idStrPtr );
        HRESULT setFacility ( HcsLogFacility* facilityPtr );
        HRESULT clearFacility ( void );
        BOOL isFacilitySet ( void );
   inline BOOL isTapped ( void ) { return( myIsTapped ); };
   inline void setTapOn ( void ) { myIsTapped = TRUE; };
   inline void setTapOff ( void ) { myIsTapped = FALSE; };


        virtual ~HcsLogFacilityPort ( void );
protected:
        /* defaults to writing errors only to the NT SystemLog (chicago??)
unless an HcsLogFacility is made available
                and is functioning */
        void dumpRec ( const Asn1ObjectId* typePtr, char* timeAsn1StrPtr,
```

```
char* idStrPtr, char* recordPtr ) ;

private:
        HcsLogFacilityPort ( void ) ;
        void* operator new ( size_t size ) ;
private:
        HcsLogFacilityOlePtr    myFacOlePtr;
        BOOL                    myIsFacSet;
   BOOL                 myIsTapped;
};
```

Effects on the HcsResourceImp Interface

HcsResourceImp::putLogRecord implementation: keeps and internal HcsLogFacilityPort instance which is used to forward records through. Before a record is actually forwarded, HcsResourceImp checks to make sure an HcsLogFacility has been assign to its port. If this is not the case, an attempt is made to locate the local HcsLogFacility for that node via the HcsResrcBusIf::locateLocalResourceById method (new). In any case, the log record is written to the via the resource instance's HcsLogFacilityPort.

New: The HcsResource interface defines, and the HcsResourceImp implements,

two methods:

HRESULT enableLogging ([in, string]char* forLogRecClassPtr);

HRESULT disableLogging ([in, string]char* forLogRecClassPtr);

These are used by support to enable and disable logging dynamically within instances of HcsResources.

The HcsLogFacility class and HcsLogFacilityPorts

Notice that any HcsLogFacility can be referenced by a HcsLogFacilityPort. This means that some components may wish to connect directly to the CLF. A good example of this may be the Resource Bus Manager (RBMGR.EXE).